Originally Published: Thursday, 5 October 2000 XP-002395974 Author: Chris Campbe
Published to: enhance_articles_sysadmin/Sysadmin Page: 1/1 - [Printable

Linux and Windows NT 4.0: Basic Administration - Part III

Back-ups are a crucial part of server administration, and ironically enough, the most typical point of failure. The typical cause of failure is generally the fault of poor administration. The administrator sets the back-ups to run and doesn't check their progress. All too often, the failure of a back-up is discovered when an attempt is made to recover.

Back-up and Retrieval

Back-ups are a crucial part of server administration, and ironically enough, the most typical point of failure. The typical cause of failure is generally the fault of poor administration. The administrator sets the back-ups to run and doesn't check their progress. All too often, the failure of a back-up is discovered when an attempt is made to recover.

There are many strategies when planning back-ups, the pros and cons of which are all debatable. We will instead focus here on the methods used to backup.

Windows NT still has the backup command from its DOS ancestor, and for the most part, it is suggested that this is left with DOS in the dark and dusty annals of computer history. This program is limited to drive to drive copying and does not really count on using tape drives, unless the tape drive were to be somehow mounted as a letter, an occurance which does not occur in Windows NT.

Tape Drives are seen as devices, and to this end, Windows NT comes with the NTBackup utility. It is accessable by:

START -> PROGRAMS -> ADMINISTRATIVE TOOLS -> BACKUP

Here, drives, directories and files can be selected, then 'Backup' launched. A tape is selected and the back-up will run. Restoring files works in the opposite, with the drives, directories and files being selected off the tape and then the point of restore on the local and attached drives being selected. The GUI for this is nice and easy, and pretty quick for a GUI back-up.There are other (typically slower) third-party back-up solutions availabe, but they will not be covered here. From single DAT drives, DLTs to terrabyte plus robot controlled back-ups are a large and diverse subject where your configuration should be tailored to your needs. Here we will cover just a simple back-up with a single DAT.

The GUI is indeed very lovely, but there is one essential problem with it. There is no scheduler, so the back-ups would be initialized manually each time. Despite rumours that computer people have no lives, this still just won't do. Luckily, we have another option.

Back at the command prompt again:

```
c:\> ntbackup {backup } {drives} /D{label} /B /HC:{on } /T
{differential} /L {logfile} /Tape:{1}
restore off full
eject incremental
```

/B - Specifies Registry to be backed up
/Tape - Specifies Installed tape unit if more than one

Restore, obviously, would be the same function but with restore as the first variable.

We can write a batch script to do our backups:

```
@echo off

net use h: /d
net use i: /d
net use j: /d
net use k: /d
net use h: \\server_02\c$
net use i: \\server_02\d$
net use j: \\server_03\c$
net use k: \\server_03\d$
ntbackup backup c: d: h: i: j: k: /D"Daily Backup" /B /HC:on /T full /L c:\temp\logs\daily.log
ntbackup eject
cat daily.log >> weekly.log
```

Here we can back-up the full contents of the local and remote servers, log it to daily.log, eject the DAT tape and append the log to a weekly log. The last command assumes that Resource Kit has been installed. Additional scripting may be done to rename the log files by date, etc.

We have a script to execute the back-up, so now we need to schedule it. Windows NT 4.0 has a task scheduling service; it is installed but is not active by default and must be activated:

START-> SETTINGS -> CONTROL PANEL

Click 'Services' and then 'Task Scheduler.'

Change Startup to 'Automatic.' Click 'OK' and then click 'Start.'

With the scheduler now running, we can add the task by typing:

```
AT {Server(if remote)} {time} /EVERY:{date} {command} (id) (/delete)
/NEXT:{date/#}
```

id - Is an identification number assigned to a scheduled command. */delete* - Cancels a scheduled command. If id is omitted, all the scheduled commands on the computer are canceled. *time* - Specifies the time when command is to run. */every:date[,...]* - Runs the command on each specified day(s) of the week or month. If date is omitted, the current day of the month is assumed. *"command"* - Is the Windows NT command, or batch program to be run.

So, for instance, to run a backup called backup.bat every monday at 2am:

```
c:\> at 2:00 /every: Monday backup.bat
```

The scheduler may also be altered using a GUI that comes with (you guessed it) Resource Kit!

START -> PROGRAMS -> RESOURCE KIT 4.0 -> CONFIGURATION -> COMMAND SCHEDULER

Here, new jobs may be added, rescheduled etc. Unfortunately, many instances have occured where this interface is not reliable, causing tasks never to run, so this interface is not

particularly advised.

Another very important item in Windows Back-up and restore is to back-up the registry. There are several different ways to do this:

```
c:\> rdisk /s
```

This will launch a GUI interface to create the rdisk.

```
c:\> regback {locationpath} - Backup registry to location on
disks/remote disks.
c:\> regrest {locationpath} - Restore registry from location on
disks/remote disks.
c:\> savekey - Save specific registry key
c:\> restkey - Restore specific registry key
```

These tools are available in Windows NT 4.0 Resource Kit, and may also be used to back-up the registry. Again, it is important to mention that all of the password information is stored in a hash file, which is included when a registry backup is created. A copy of the registry in the wrong hands can be easily cracked for passwords. (http://www.lopht.com/l0phtcrack/)

It is also a good idea to get a copy of the boot sector in case of file corruption. c:\boot.ini has the Windows NT boot menu in it, and should it get damaged or erased, the system will not boot. The inside of the file will look something like:

```
[boot loader]
timeout=30
default=multi(0)disk(0)rdisk(0)partition(1)\WINNT
[operating systems]
multi(0)disk(0)rdisk(0)partition(1)\WINNT="Windows NT Server Version
4.00"
multi(0)disk(0)rdisk(0)partition(1)\WINNT="Windows NT Server Version
4.00 [VGA mode]" /basevideo /sos
```

If file permissions are changed;

```
c:\> attrib boot.ini -R -A -S -H
```

This file may be edited to add and remove items form the menu, change, timeout, etc.

So, to create a Windows NT boot disk:

Format a floppy. (You must do the format from Windows NT)

In Windows explorer or "My Computer" go to "View" then to "Options", "View" and click on "Show All file Types". This will configure Windows NT Explorer to be able to view hidden and system files. Now copy the following files from c:\ (or your boot drive) to the floppy disk:

```
NTLDR
BOOT.INI
NTDETECT.COM
BOOTSECT.DOS
NTBOOTDD.SYS
```

If the Windows NT Server ever has an issue booting, this disk should boot it successfully.

Althought 3rd party products are available in Linux, as are often used in Windows NT, Tape Archive and Restory (*tar*) is a tool that comes with Linux and is the most common tool used fo general backups.

```
#tar {options/functions}f {filename(s)}
```

c - Create new archive
x - Extract from archive
t - List contents of archive
r - add files to archive
u - update files in archive
v - verbose
k - do not overwrite existing files
f - filename
T - use contents of a filename

tar, incidentally, can also function as a compressor/decompressor compatible with pkunzip/winzip in Windows. Syntax:

```
#tar zxf {zipfile}
```

Other options for zipping/unzipping are gzip and gunzip.

Backups are scripted in shell scripts in Linux. Shell scripts are very similar to it's DOS or Windows based relative, the batch file. An interesting excpetion being in Linux extensions are irrelevant. A shell script can be called anything, what matters is that the file is declared executable. In Linux, this is handled with the chmod command (change mode).

```
#chmod 777 {filename}
```

would be equivalent to

```
c:\> attrib -R -A -S -H {filename}
```

In DOS.

The 777 represents octal permission levels for the file. This is a bit lengthy and off of the subject, so for more details, type:

```
#man chmod
```

in Linux.

tar, when used for backups, writes directly to the tape drive by its system device name. Any drive device could technically be used, but tapes are most common. A backup command would look like:

```
#tar cvf {tape_device} {Directories to backup}
```

such as:

```
#tar cvf /dev/nrst0 /etc /home
```

And to restore:

```
#tar xvf {tape_device}
```

such as:

```
#tar xvf /dev/nrst0
```

This would just be a basic backup and restore. *tar* can take lists of files instead of just directories also. If a list of files and directories to be backed-up were put into the file /etc/archive.list, *tar* could read it and back to tape drive, such as here to /dev/nrst0:

```
#tar -cv -T /etc/archive.list -f /dev/nrst0
```

Further scripting with this could produce incremental backups:

```
#find / -mtime -7 -print > /etc/archive.list #tar -cv -
T /etc/archive.list -f /dev/nrst0
```

To further control the tape device, we will use the *mt* command:

```
#mt {device} {rewind}
reten (rentension tape)
fsf 1 (Forward Skip File-skips 1 file)
fsf 2 (Forward Skip File-skips 2 files)
```

Once a TAR command completes, the tape goes back to the beginning of the file marker. After reading one file on the tape, type:

```
#mt {device} fsf 1
```

to go on to the next file.

In Linux, the device files also control activity. The specific tape device we have used here, /dev/nrst0, is non-rewinding tape 0. Others are available, such as:

```
/dev/nrft0 /dev/nrft1 /dev/nrst0 /dev/nrst1
```

We chose not to use rewinding drivers, as we are controlling those sorts of fucntions with *mt*.

Scheduling in Linux is at the hand of the cron (chronological) service daemon. This service keeps all assigned tasks in a crontab file; there is a file for each user who uses crontab. These files are located in the /var/spool/cron/crontabs directory.

To add a job, edit the crontab:

```
#crontab -e
```

This will open the crontab file in *vi*. The format of the records are:

```
MN HR DM MO DW {command}
```

where:

MN = Minute
HR = Hour
DM = Day of the Month
MO = Month
DW = Day of the Week

Any non-valid fields should be kept with an *. To schedule backup.sh to run a 1:30 AM every Saturday, for example:

```
# Execute Weekly Full Backup
30 1 * * * sat /usr/backup.sh
```

The comment field is forwarded by a #, so that the text on the line is ignored.

Finally, in case something unfortunate should happen to the boot sector in Linux, it can be written to disk for backup:

Backup MBR (Linux):

```
#mount /dev/fd0 /mnt/floppy
#dd if=/dev/hda of=/mnt/floppy/mbr bs=512 count=1
#umount /dev/fd0
```

Restore MBR (Linux):

```
#mount /dev/fd0 /mnt/floppy
#dd if=/mnt/floppy/mbr of=/dev/hda bs=512 count=1
#umount /dev/fd0
```

Shell scripting, like batch programming, is a skill that grows as an administrator's prowess grows. As advanced as a GUI may feel, when it is time to get down and dirty, the commandline is often. Even in Windows NT, tasks like login scripts require a degree of ability in command line skill. An administrator not capable of shell work is definitely not using their server, whether it be Windows NT or Linux, to its fullest ability.